1-1997

Integrated queries to existing bibliographic and structured databases

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

Ying LU
Nanyang Technological University

DOI: https://doi.org/10.1006/jnca.1996.0036

Journal of Network and Computer Applications (1997) **20,** 3–24
https://doi.org/10.1006/jnca.1996.0036

Integrated queries to existing bibliographic and structured databases

Ee-Peng Lim and Ying Lu*
School of Applied Science, Nanyang Technological University, Nanyang Avenue, Singapore 639798

It is widely accepted that future digital library applications have to be built upon different kinds of database servers to draw upon different forms of data, including bibliographic, text, multimedia, and structured data. In this paper, the problem of integrating existing public bibliographic databases and structured databases which reside at different locations in the network is addressed. Although bibliographic data is semistructured, its attribute set is often determined by an international standard known as MARC. To unify bibliographic and structured data, the well-known SQL was extended to model bibliographic related attributes and queries. In particular, a new data type was added to model attributes in the bibliographic database. Specialized functions have also been designed that can be used to formulate queries on bibliographic data, as well as queries on both bibliographic and structured data. By combining the SQL extensions with other query language extensions that access general text data, it is believed that a versatile query language layer can be made available for future digital library application development.

1. Introduction

Traditionally, public library systems operate in isolated environments. Although there is an increasing number of automated library systems available from the network, most of these systems are still standalone—they do not cooperate in inter-library activities such as *inter-library loans* and *concurrent library searches*. A lack of integration also exists between library systems and other kinds of information resources such as document databases (multimedia or text) and structured databases. Examples of document databases include CD-ROM text databases and WAIS databases [1].

Clearly, to build advanced digital library systems and applications, one has to integrate these different kinds of databases together and to provide value-added search and retrieval services over them [2]. For example, the future digital library should allow users to perform online-catalogue searches followed immediately by retrieving the documents associated with the selected catalogue records. Users should also be allowed to query library catalogues and their own structured databases in an integrated manner.

This paper focuses on the integration of bibliographic databases and structured databases which may reside at different locations in the network. The authors believe that the bibliographic databases maintained by almost all the public library systems represent an important source of information that can be shared among library users. Unlike documents, bibliographic data are not usually subjected to copyright restriction,

Correspondence to: Ee-Peng Lim, School of Applied Science, Nanyang Technological University, Nanyang Avenue, Singapore 639798. Email: aseplim@ntuix.ntu.ac.sg.

and future library users may always have to search bibliographic databases first before they can decide on the actual documents to acquire. Bibliographic databases maintained by the library systems are also of high quality since they are usually created and managed by professional cataloguers. On the other hand, as SQL has become the de-facto standard for structured databases and SQL database engines are becoming popular, it is anticipated that there will be a growth in the number of structured database users and databases related to digital libraries. For example, holding information about borrowed and reserved books/periodicals in the public libraries may be maintained in SQL databases in the future. Publishers and bookstores may also use SQL databases to store information about their books.

To support queries on bibliographic databases as well as queries on both bibliographic and structured relational databases, the relational model has been extended so that remote bibliographic databases can be modeled as tables, and tables can include attributes from both bibliographic and structured databases. Some bibliographic related predicates/functions to the SQL language [3] have also been added so that bibliographic attributes can be manipulated.

This integrative approach allows both bibliographic and structured relational databases to co-exist while preserving the autonomy of their database servers and legacy applications. To homogenize the disparate interfaces to existing bibliographic databases, the Z39.50 standard protocol [4] was adopted to search and retrieve bibliographic data. The bibliographic data retrieved are represented in the populat MARC (MAchine-Readable Cataloguing) [5] data exchange and storage format.

1.1 *Motivations*

Bibliographic data usually demonstrate weak database schema structures (see Section 2), and can be classified as semi-structured data. Semi-structured data can also include formatted text data found in documents such as SGML (Standard Generalized Markup Language) [6] and ODA (Office Document Architecture) [7] documents. In contrast, unformatted text, image, audio or video data are usually classified as unstructured data. Studies on the integration of general semi-structured data and structured data have been reported in [8–10]. They typically focus on integrating relational data with text data which can be represented as labelled graphs, or text data written in SGML. Unfortunately, these proposed schemes may not represent bibliographic data well, since the latter have been stored in a more restrictive structure determined by the international MARC standard. In this paper, we have chosen to extend SQL due to its versatility and popularity among structured database users.

In fact, this extended SQL can be viewed as part of an integrated digital library service layer upon which future digital library applications can be built. This service layer encapsulates the complexity of accessing different kinds of remote databases. To the application builders, the locations and operational differences between the database servers are made transparent. Using an unified query language, one can access different forms of data within a single query. Moreover, with the additional functional layer handling the optimization of multi-database queries, the library applications can expect queries to be evaluated more efficiently. This layered approach to building digital library applications is shown in Fig. 1.

Although advanced digital library applications can also be built directly upon the

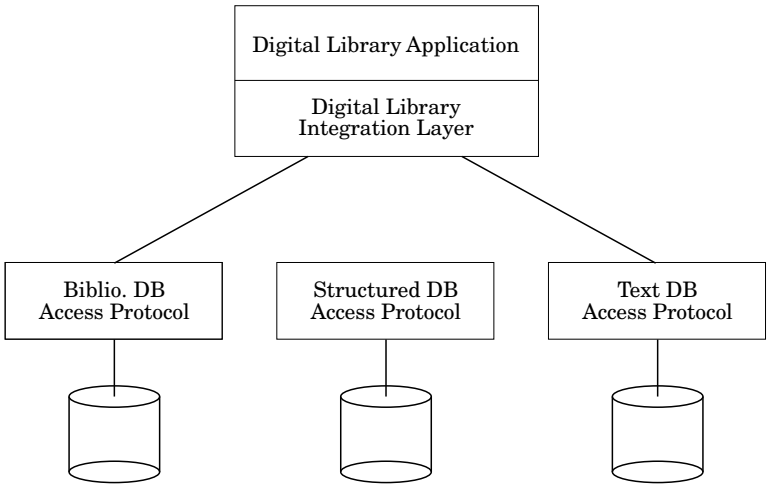

Figure 1. Layered approach for developing digital library applications.

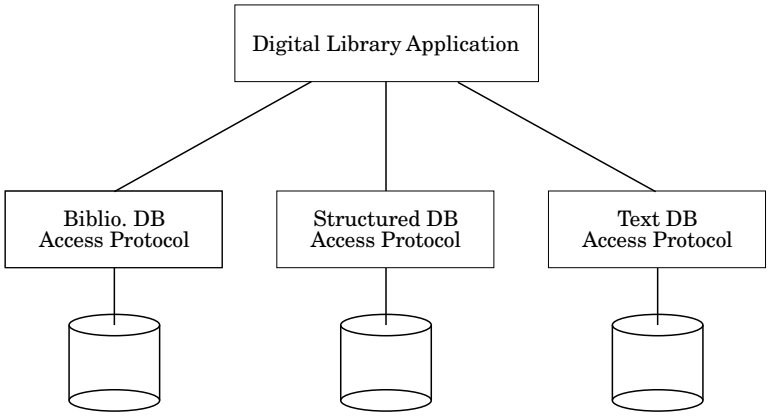

Figure 2. Monolithic approach for developing digital library applications.

various network protocols without an integrated date model and query language, locating and querying different bibliographic and structured databases have to be handled within the digital library applications themselves. This is also known as the monolithic approach as shown in Fig. 2. The major disadvantage of the monolithic approach is that it hinders the development of integrated digital library applications by requiring application builders to resolve the integration issues whenever an application is developed.

1.2 *Related works*

In the following, we briefly survey related research:

- TSIMMIS Project: in the area of modeling and querying semi-structured data,

Papakonstantinou *et al*. proposed an object exchange data model called OEM (Object Exchange Model) [9]. In OEM, databases do not have the concept of schema which has traditionally been used to capture the structure of a set of data objects. Instead, every object is represented by a 3-tuple <*identifier, label, value*> where *identifier* is the object ID, *label* is analogous to the attribute name, and *value* represents the corresponding attribute value. To query a semi-structured database which consists of a set of 3-tuples representing objects, a query language known as LOREL has been defined [11]. LOREL is very different from the familial SQL as it handles missing data, missing attributes, type mismatches, heterogeneous sets etc.

Since OEM and LOREL do not resemble the familiar relational model and SQL respectively, they will pose difficulties for the application developers. Without the schema concept, it may be difficult for existing structured database users to query their data. Although structured databases can also be modeled by OEM and be queried using the LOREL language,* the amount of redundant information caused by storing the structured data in the semi-structured way can be wasteful.

- T/RDBMS Project: instead of designing an entirely new data model and query language, Blake *et al*. proposed SQL extensions to accommodate text data in the structured databases [10]. In their extended SQL, a special TEXT attribute is introduced to represent text together with its grammar written in a SGML DTD. Special functions and search predicates that operate on TEXT attributes have been proposed. Nevertheless, it may be an overkill to store bibliographic data as TEXT attributes since there is no complex grammar for bibliographic data. Furthermore, the attributes found in most bibliographic databases have been governed by the MARC standard.
- Multidatabase systems: our problem of integrating existing bibliographic and structured databases is related to the area of multidatabase research [12,13]. Multidatabase systems aims to integrate the schemas of multiple existing structured databases together and to support queries against the combined global schema. Integrated digital libraries can be treated as a special kind of multidatabase system since both involve multiple databases and both must preserve the local autonomy. However, in some say, our integrated digital library environment is more complex than that of multidatabase systems because semistructured bibliographic databases are involved.

1.3 *Motivating examples*

In this section, an example is given that involves structured and bibliographic databases, and some application scenarios in order to illustrate our integration approach.

Let `RefDB` and `PubDB` be two structured databases residing at different locations. `RefDB` is a reference database which contains the reference information related to a researcher's interest. `RefDB` consists of a table `RefTB`, whose schema is shown in Fig.

* This can be done by treating structured data as semi-structured data.

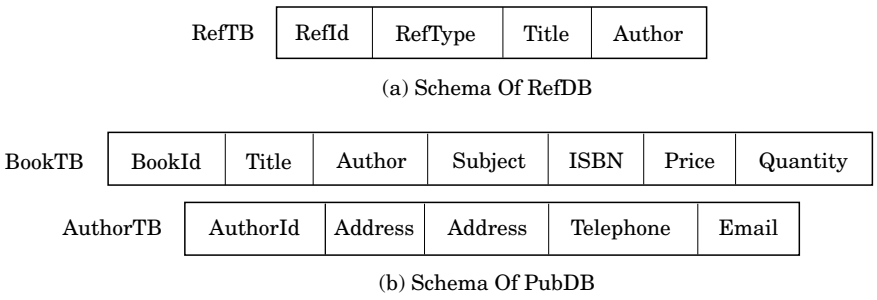

Figure 3. Schemas of example database.

3(a). `RefID` is the identifier for the reference records. `RefType` indicates the type of the reference which can be either a book or a journal. `Title` and `Author` are the other columns found in the reference table. `PubDB` is a database owned by a publisher. It consists of two tables: `BookTB` and `AuthorTB` (see Fig. 3(b)). `BookTB` contains the information of all books published by the publisher. Every book has a unique identifier `BookId` and some bibliographic information such as title, author, etc. The table also maintains the price and quantity information. `AuthorTB` contains the particulars of the authors whom the publisher has commissioned. Each author has a unique identifier `AuthorId` and some personal information including name, address, telephone number and email address.

Other than the two structured databases, this example includes bibliographic databases found in the NTU (Nanyang Technological University) library and the NUS (National University of Singapore) library. These two bibliographic databases are in the MARC format. The attributes of MARC databases will be described in Section 2.1.

When these databases are not integrated together, it can be both awkward and time consuming to carry out tasks involving them. Consider the following scenarios:

- Scenario 1: when the owner of `RefDB` wants to check if their references can be found in the NTU library, they have to manually search against the MARC database for each `RefTB` record. This is undesirable when `RefTB` is very large. In this case, if a join operation between structured and bibliographic databases is supported, the above query can be expressed easily.
- Scenario 2: to search the NTU and NUS libraries for a book with a known title, one has to log into the two library systems and perform identical searches against their MARC databases. This process can be painstaking when logging into different library systems and searching their bibliographic databases can only be performed sequentially.
- Scenario 3: to find books that exist in `BookTB` but not in the NTU library, one has to search against the NTU library for each book listed in `BookTB`. This query can be easily expressed if set difference operations between these two databases are allowed.
- Scenario 4: to avoid repeated searches, one may like to store the results of previous searches, and to be able to use them for future queries (e.g. results of queries mentioned in the above three scenarios). In this case, it is necessary to store the

Table 1. *Popular MARC fields*

Tag number	Field name
001	Control number
005	Data and time of latest transaction
008	Fixed length coded data elements (date entered on file, etc.)
020	International Standard Book Number (ISBN)
040	Cataloging Source
100	Main entry—personal name
110	Main entry—corporate name
111	Main entry—conference or meeting
245	Title statement
250	Edition statement
300	Physical Description
500	General note
600	Subject added entry—personal name
610	Subject added entry—corporate name
650	Subject added entry—topical heading
700	Added entry–personal name

results as tables which may contain a mixture of sructured and bibliographic attributes.

2. Local bibliographic database and query model

To design our integrated data model, we must consider the various standards governing the interfaces to the existing bibliographic and structured databases. These standards ensure that the integrated data model can actually be realized by re-using the tools and utilities provided by library system vendors. While SQL has been widely accepted as the de-facto data model and quety language to access pre-existing structured databases [14,15]*, there is correspondingly a popular *suite* of data exchange formats and a query interface for pre-existing bibliographic databases. They are known as MARC (MAchine-Readable Cataloguing) [5] and Z39.50 [4], respectively.

2.1 *MARC Standard*

Unlike a relational database which may include several tables, each with a different scheme, most bibliographic databases in public libraries consist of simply MARC records. MARC is a set of standards that describes how cataloging information can be stored or exchanged. It defines a comprehensive set of fields that describe library material including books, periodicals, films, maps, sound recordings, etc. Examples of such fields include control number, ISBN, title statement, main entry-personal name, etc. Every field is assigned a tag so that it can be manipulated by library software. The most popular MARC field tags are shown in Table 1.

* Here, we base our assumption on the fact that most BD vendors either have relational databases or are in the process of making pre-existing databases look relational so that these may be queried through SQL.

The book: Senn, James A. Information technology in businness: principles, practices, and opportunities. Annotated instructor's ed.

```
001     AAS-5906
005     19950106105505.2
008     940422s1995     njua     b     00100 eng
010     a     94017259   $o     94017259   $
035     a (SILAS)7061013 $
020     a 0134849086 (Instructor's ed.) $
020     a 0134843045 (Student ed.) $
040     a DLC $c DLC $d DLC $
050 00 a HF5548.2 $b .S4366 1995 $
082 00 a 650/.0285 $2 20 $
092     a HF5548.2.S478 $
100 10 a Senn, James A. $
245 10 a Information technology in business : $b principles, practices,
        and opportunities / $c James A. Senn. $
250     a Annotated instructor's ed. $
260 0   a Englewood Cliffs, N.J. : $b Prentice Hall, $c c1995. $
300     a xxiv, 598, 2, 16 p. : $b col. ill. ; $c 26 cm. $
504     a Includes bibliographical references and indexes. $
650   0 a Business $x Data processing. $
650   0 a Information storage and retrieval systems $x Business. $
650   0 a Information technology. $
650   0 a Local area networks (Computer networks) $
```

Figure 4. A bibliographic record example formatted in MARC.

Being a bibliographic exchange standard, MARC facilitates the shared cataloging process. Among the various MARC standards, we have chosen to model USMARC closely since it is the most popular MARC variant and there is a trend of other MARC variants converging towards USMARC. For convenience, we shall use MARC and USMARC interchangeably henceforth.

Figure 4 shows a bibliographic record and its corresponding MARC record.* The MARC record consists of fields which are variable length strings. Each field is given a tag with a specific meaning. For example, the field with tag 001 is the control number, and the field with tag 100 is a main entry personal name, which is also the author's name. Most fields are subdivided into subfields to define individual elements within the fields and to attach more refined meaning to the subfields. Each subfield is assigned a subtag (e.g. '$a', '$b', '$c', etc.) which is unique within the field. In Fig. 4, field 260, which contains publication information, has three subfields: '$a', '$b' and '$c', representing the place of the publication, the name of the publisher and date of the

* We have intentionally left out the preamble and directory information to make explanation simpler. The $'s are used as delimiter symbols.

Table 2. *Comparisons of MARC databases and relational databases*

MARC database	SWL relational database
MARC fields contain variable length strings as values.	RDBMS supports integer, fixed-length character string, and floating-point numbers. Variable length string with no maximum is not supported in SQL standard.
Same field may appear more than once in a MARC record.	An attribute can only appear once in a record.
Different subsets of MARC fields can be found in different MARC records.	All records in a table have the same set of attributes. Their attributes are determined by the table scheme.
A MARC field may consist of multiple subfields.	Each field has atomic data values.
A MARC database consists of only a set of records.	Multiple tables can exist in a relational data base.

publication, respectively. Unlike tuples in a relational table, a MARC record usually contains only a subset of all attributes defined by MARC. The attributes included in a MARC record depend on the type of bibliographic record (book, serial, film, audio recording, etc.). Hence, we say that MARC databases consist of variable fields. Furthermore, the same field may appear more than once (e.g. tag 650 in Fig. 4).

In Table 2, we summarize the differences between MARC databases and SQL databases.

2.2 *Z39.50 information retrieval protocol*

While MARC is a well-accepted standard to store and exchange bibliographic information, without an information retrieval protol one cannot access the MARC records keps in existing libraries. A strng effort in standardizing remote library retrievals has resulted in the ANSI Z39.50 protocol [4]. Z39.50, likes its counterpart RDA (Remote Data base Access) [14] in the relational database domain, has been implemented at many libraries (including Data Research Public Library, AT&T Research Library and Nanyang Technological University Library) and most library software vendors are quick in adopting it. As depicted in Fig. 5, a library system in the Internet environment can support Z39.50 by operating a server daemon which listens to incoming query requests at some port address.* A Z39.50 server can usually manage one or more bibliographic databases in the MARC format. Multiple Z39.50 clients can communicate with the same server simultaneously, but only one server can be accessed by a Z39.50 client at any time. A Z39.50 client can specify in its query request the specific bibliographic databases to be queried, thus broadcasting the same query request to these databases.

Queries to bibliographic databases managed by Z39.50 servers have to be specified as Z39.50 search requests. As a result, the way we model and query the remote bibliographic databases has to be constrained by what the Z39.50 search service can

* TCP Port 210 has been assigned to Z39.50 by the Internet Assigned Number Authority.

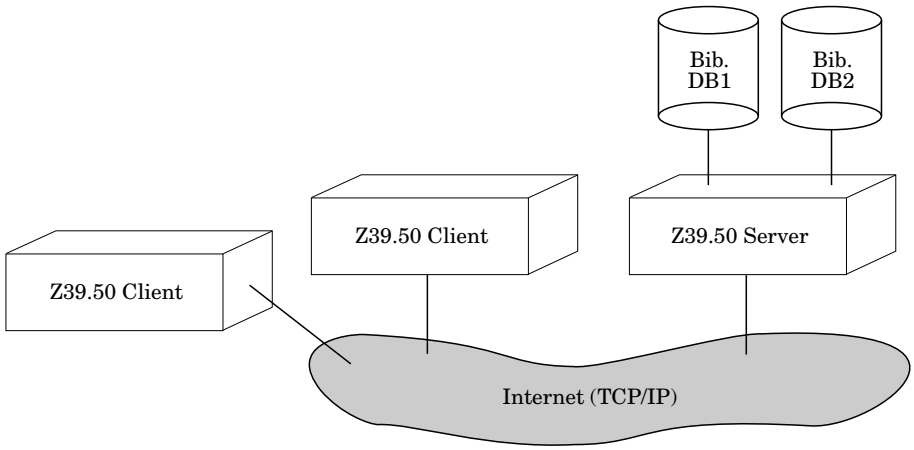

Figure 5. Z39.50 client-server architecture.

provide. The Z39.50 search service allows queries of five different types to be specified. Nevertheless, Z39.50 only mandates one of the query types, namely the Reverse Polish Notation (RPN) query type*, to be supported by all Z39.50 servers. In this paper, our discussion will be restricted to RPN queries. The essential features of RPN queries are:

- Bib-1 attribute set for specifying search predicates: Z39.50 is designed to support a variety of bibliographic databases (MARC or non-MARC). Hence, different attribute sets† can be used to specify search predicates within RPN type queries. Among them, the attribute set Bib-1 has been widely used (see Appendix A). Note that each attribute included in Bib-1 corresponds to one or more MARC fields. In other words, attributes involved in search predicates have to be mapped into MARC fields when queries are evaluated against the MARC databases. As almost all servers currently support only the MARC standard for interchange of records, this paper will only focus on query results represented in MARC format. It is however possible to extend the approach to accommodate other representations, such as ASN.1 [16]. No matter which representation is used to represent a query result, Bib-1 attributes in the search predicates will nt be found in the result set.
- Z39.50 client can query only one Z39.50 server at a time: A Z39.50 client, at one time, can only interact with one Z39.50 server. The client may query one or more bibliographic databases managed by the same server. If a RPN query is to be broadcast to databases located at different sites, multiple clients will be required.
- Expressive power of RPN query: RPN queries support string related comparisons and the usual boolean operators (AND, OR, NOT). However, all binary algebraic operators, such as join, intersection, union, etc., are not available. Aggregation and nested queries are also not supported.

* Also known as the type-1 query.

† The attribute set here refers to more than what is commonly known as a set of attributes. An attribute set includes a set of encoded attribute names together with additional usage semantics [4].

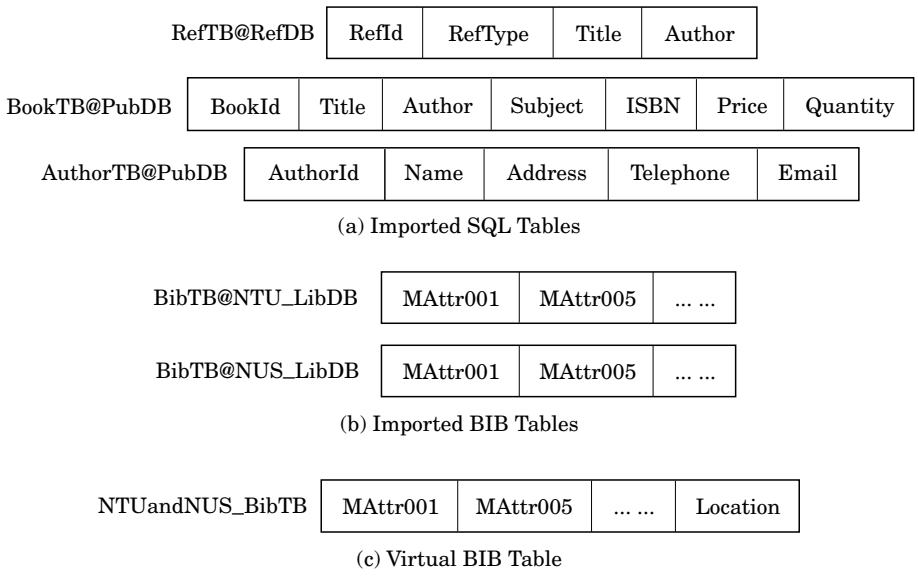

Figure 6. Imported tables of the motivating example.

3. Integrating the bibliographic data model with the relational model

In this section, we present an integrated data model based on an extension of the SQL relational model. The extensions include: (1) MARC-attribute and MarcString; (2) virtual bibliographic table; (3) Bib-1 attribute and (4) alias-attribute. Our proposed integration is intended to be loosely coupled. Hence, existing bibliographic databases are modeled as relational tables in the integrated data model just like relational tables from other existing relational databases. This loosely coupled approach has also been adopted by the Multidatabase project [12] which integrates structured databases. Furthermore, the integration does not require any modification to the existing databases thus upholding their local autonomy.

Local databases and their tables have to be imported into the integrated database before they can be queried. The importation can be done by the `IMPORT DATA BASE` and `IMPORT TABLE` commands. Suppose it is known that `RefDB` in our motivating example is maintained by a SQL server located at `sentosa.sas.ntu.ac.sg`. If the server provides query services via port 210, one can then import `RefDB` and its `RefTB` by:

```
IMPORT SQL DATA BASE RefDB (IP=SENTOSA.SAS.NTU.AC.SG, port=210)
IMPORT SQL TABLE RefTB@RefDB (RefId int, RefType int, Title
                 char(60), Author char (40))
```

Figure 6(a) shows all SQL tables that have been imported into our integrated database. In the following, we describe the other essential features of our extended model.

3.1 *Marc-attribure and MarcString*

As mentioned in Section 2.1, MARC standard defines a set of fields describing bibliographic information. In our integrated data model, we define marc-attribute to be a set of MARC fields sharing the same tag value. Every local MARC database is thus modeled as a bibliographic table (BIB table) which consists of marc-attributes modeling the full set of MARC fields. All marc-attributes in a BIB table are of MarcString data type described by the following BNF grammar:

```
<MarcString>: (<tag>, <element>*)
<element>: ('<subtag> <sub-element>'*)
```

A MarcString value consists of a `tag` and a set of `elements`. An element consists of repeating pairs of `subtag` and `sub-element`, which correspond to the MARC subfields. Consider the MARC fields of tag 650 in Fig. 4. Four different subject added entries of topical heading appear in the record. These MARC fields are represented by a marc-attribute with MarcString value:

```
(650, ($a Business' '$x Data processing')
      ('$a Information storage and retrieval systems'
      '$x Business')
      ('$a Information technology')
      ('$a Local area networks (Computer networks)'))
```

By including the MarcString data type in our integrated data model, tables containing a mixture of marc-attributes and structured attributes can also be represented. Since MarcString values are of variable length and they consist of subelements, we have adapted some techniques originally designed for variable length records to store and manipulated them.

A marc-attribute is named `MAttr('attribute-name')` where `attribute-name` is the name assigned by the MARC standard. Marc-attributes can also be referenced by 'MAttr' followed by their tag numbers. For example, the marc-attribute whose tag is 100 can be represented by `MAttr ('main entry Ł personal name')` or `MAttr100`.

A MARC database is imported as a BIB table in our integrated database. The two BIB tables imported from our example bibliographic databases are shown in Fig. 6(b). For MARC records that do not have a full set of MARC attributes, we assign NULL values to their missing MARC attributes. Unlike importing SQL tables, the importation of BIB tables does not require schema information because all BIB tables share the same schema. For example, we can import `NTU_LibDB` as follows:

```
IMPORT BIB TABLE BibTB@NTU_LibDB (IP=LIBSERVER.NTU.AC.SG,
                                  port=210)
```

3.2 *Virtual bibliographic table*

In this integrated data model, a virtual bibliographic (BIB) table can be defined upon multiple BIB tables imported directly from local MARC databases. Each of these BIB tables will be called a member BIB table. The purpose of defining virtual BIB tables is to facilitate queries to be broadcast to multiple local BIB tables managed by different

servers. In this manner, users do not need to specify separate queries to different servers. For example, we may define NTUandNUS_BibTB as a virtual BIB table consisting of the NTU library and the NUS library information:

```
DEFINE VIRTUAL BIB TABLE NTUandNUS_BibTB AS UNION OF
        BibTB@NTU_LibDB, BibTB@NUS_LibDB
```

From the user's perspective, a virtual BIB table is just like any other imported BIB table. The user will not distinguish between querying virtual BIB tables and other BIB tables. However, a query on a virtual BIB table is broadcast to all its member BIB tables, and the results of these identical queries are unioned.

The schema of a virtual BIB table is similar to that of any imported BIB table except that it has an extra location attribute which is of string data type (see Fig. 6(c)). The attribute specifies the member BIB table from which a virtual BIB table record is derived. Location attributes can be added by a query processor whenever it obtained search results from the local MARC databases.

For example, the researcher in the motivating example may want to define the virtual BIB table NTUandNUS_BibTB if they often search the two libraries together. With the location information, they can find out which library holds the book and may subsequently send a reservation form to the appropriate library.

3.3 *Bib-1 attributes*

To support Z39.50 queries on our BIB tables, a mapping between Bib-1 attributes and marc-attributes is needed. This mapping applies to all BIB tables and is given in Appendix A. Since Bib-1 attributes can be treated as standard surrogates for marc-attributes, their data types are therefore MarcString.

Every Bib-1 attribute is named by BAttr('attribute-name') where attribute-name is the name assigned by the Z39.50 standard. Bib-1 attributes are named by 'BAttr' followed by their attribute ID's. For example, Bib-1 attribute 1003 can be represented by BAttr('Author') or BAttr1003.

Bib-1 attributes, when used in the search predicates of a query, will be translated into predicates on the underlying marc-attributes connected by OR operators. When a Bib-1 attribute is used in the SELECT clause of a query, it will be replaced by its underlying marc-attributes. For example, if Bib-1 attribute 1004 (Author-name personal) is specifed in a SELECT clause, it will be replaced by MAttr100, MAttr400, MAttr700 and MAttr800.

3.4 *Alias-attributes*

An alias-attribute represents two or more attributes that share the *same data type* and come from the *same table*. Identical search predicates are often specified on these attributes in a query. The use of the alias-attribute can be best illustrated by the keyword concept supported by most bibliographic search engines. A search on keyword corresponds to applying the same search predicate to several bibliographic attributes which are defined to constitute a keyword. Unlike Bib-1 attributes which have a fixed mapping to their underlying marc-attributes, the exact meaning of keyword is determined by the applications. Usually, the bibliographic attributes that constitute a keyword

include author, title, subject, etc. Since the keyword concept is neither supported by the Bib-1 attribute set in the current version of Z39.50 not included in the MARC standard, our integrated model allows it to be defined as an alias-attribute. For example, `keyword` can be defined as an alias for `BAttr (Title), BAttr1003 (Author)` and `BAttr21 (Subject heading)` in `BibTB@NTU_LibDB` as follows:

```
DEFINE ALIAS keyword AS BAttr4, BAttr1003, BAttr21 FOR
BibTB@NTU_LibDB
```

Like Bib-1 attributes, alias attributes can appear in both `SELECT` and `WHERE` clauses of our queries. We handle such queries in a way similar to that for Bib-1 atributes.

4. Extended query language – HarpSQL

As the relational model is extended to represent bibliographic and structured database information, it is necessary to augment the SQL query language with additional features. This extended SQL is known as HarpSQL*.

HarpSQL handles the usual SQL queries on existing structured databases†. It also supports queries that are traditionally performed by the existing applications on bibliographic databases, as well as new queries that involve both bibliographic and structured databases. The new predicates and functions that have been introduced are: (a) Contain( ): a Marc-String containment predicate; (b) Extract( ): a sub-element extraction function; and (c) MarcToText( ) and TextToMarc( ): functions to convert from MarcString to normal string and vice versa.

4.1 *Contain( ): a MarcString containment predicate*

Bibliographic information is modeled as MarcStrings in our integrated data model. Therefore, we introduce a new predicate `Contain()` for comparing an attribute of MarcString data type with a normal string. The design of `Contain()` adheres closely to the suggested Bib-1 query semantics of the Z39.50 Implementator Group [4]. A `Contain()` predicate has the format shown below.

```
Contain(attrib, search_term, [search_mode])
```

The `attrib` argument specifies the attribute to be searched. This attribute must be of MarcString data type (including Bib-1 attribute). `search_term` can be an attribute of string data type or simply a string constant. `search_mode` specifies the detailed search semantics to be used to evaluate the MarcString containment predicate. The `Contain()` predicate returns `TRUE` when the value of `search_term` is contained in `attrib` according to the desired `search_mode`, and returns FALSE otherwise.

`search_mode` is a quadruple of `sub-modes` represented by:

```
<position, structure, truncation, completeness>
```

- `position` specifies the location of the search term within the attribute in which

* The name HarpSQL is chosen because this extended query language has been adopted by an integrated digital library project called HARP.

† Assuming that the local structured database systems are either SQL-based or support SQL gateways.

it appears. For example, FIRST_IN_ELEMENT means that the search term must be the first data in any element of a MarcString.

- `structure` specifies the type of the search term. For example, IS_PHRASE requires the search term to be treated as a phrase with respect to order and adjacency.
- `truncation` specifies whether one or more characters may be omitted in matching. For example, RIGHT_TRUNC indicates that the last word of the search term may be truncated.
- `completeness` specifies whether the content of the search term represents a complete or incomplete element/sub-element. For example, COMPLETE_SUBELEMENT means that no words other than those in the search term should appear in the sub-element of the MarcString in which the search term appears.

A complete definition of these sub-modes is given in Appendix B. If a search mode is replaced by `NULL`, a default value for that sub-mode will be used. For example, if one wants to search the NTU library for books with subject containing 'distributed database' as a phrase, one may use

```
Contain(BAttr('Subject heading'), 'distributed database', <NULL,
                    IS_PHRASE, NULL, NULL>)
```

4.2 *Extract( ): a sub-element extraction function*

To allow users to extract specific sub-element values from marc-attributes, we introduce the `Extract` function:

```
Extract(attrib, subtag, num, len)
```

The `Extract` function returns a normal string which is a concatenation of all sub-elements in a MarcString sharing a common subtag, `attrib` refers to the marc-attribute to be extracted. `subtag` is the subtag of the specified sub-element such as '$a'. Since a MarcString value can have multiple elements, the integer `num` is used to specify the number of elements from which sub-elements are extracted. If `num` is 0, sub-elements will be extracted from all the elements and concatenated to one normal string. Otherwise the sub-element will only be extracted from the first `num` elements of `attrib`. `len` specifies the maximum length of the return string.

For example, MARC attribute 650 (Subject added entry—topic heading) represents subject information. An element of the attribute value may contain several sub-elements such as $a for topical heading/place, $x for general subject subdivisions, etc. To concatenate all topical heading/place information into a string of at most 256 characters, `Extract(MAttr650, '$a', 0, 256)` is used. If only the first three elements are required, `Extract(MAttr650, '$a', 3, 256)` is used instead. The results of applying these two `Extract()` functions on the MarcString value given in Section 3.1 are:

```
Extract (MAttr650, '$a', 0, 256) =
'Business; Information storage and retrieval systems;
Information technology; Local area networks (Computer
```

```
  networks)'
Extract(MAttr650, '$a', 3, 256) =
'Business; Information storage and retrieval systems;
Information technology'
```

4.3 *MarcToText and TextToMarc: conversion between MarcString and normal string*

In a query, users may want to convert some marc-attributes into normal strings. For example, the publisher in our motivating example may want to get the subject information of the books from `BookTB@NTU_LibDB` and stores them as normal strings. In this case,

```
MarcToText(attrib, len)
```

is the conversion function they may need. `attrib` indicates the marc-attribute to be converted. `len` specifies the maximum length of the return value. Unlike Extract( ) and MarcToText( ), the function first concatenates all sub-elements of each element into a normal string before it concatenates the strings of all elements together.* Given the marc-attribute for tag 650 in Section 3.1, `MarcToText(MAttr650, 256)` returns `'Business, Data processing, Information storage and retrieval systems, Business, Information technology, Local area networks (Computer networks)'`.

In contrast, `TextToMarc()` is used to add a normal string to a marc-attribute. The syntax for this function is:

```
TextToMarc(attrib, element, subtag, subelement)
```

where `attrib` is a marc-attribute. `subelement` specifies the normal string attribute or string constant to be added to `attrib`. `subtag` species the subtag for `subelement`. Since `attrib` can have several elements, an integer `element` value is required to specify the element to which `subelement` will be added. For example, to add 'Management' to the third element of marc-attribute `MAttr650` with subtag '$x', one may specify `TextToMarc(MAttr650, 3, '$x', 'Business')`.

4.4 *Query examples of HarpSQL*

In the following, we give some query examples to illustrate the features of HarpSQL.

Q1: Retrieve the title statements and author names of the books with the `keyword` phrase 'distributed database' from the NTU library.

```
SELECT  MarcToText(MAttr245, 256), Extract(MAttr100, '$a', 0,
        256)
FROM    BibTB@NTU_LibDB
WHERE   Contain(keyword, 'distributed database',
<ANY_POSITION, IS_PHRASE, NULL, NULL>)
```

We assume that `keyword` is defined as an alias for `BAttr1003(Author)`,

* Two strings are concatenated with a comma between them. All subtags are discarded.

BAttr4(Title) and BAttr21(Subject heading). The search term 'distributed database' is to be used as a phrase and can appear in any position in the three Bib-1 attributes. Hence, the predicate will be translated to:

```
Contain(BAttr1003, 'distributed database, <ANY_POSITION,
IS_PHRASE, NULL, NULL>)
OR
Contain(BAttr4, 'distributed database', <ANY_POSITION,
IS_PHRASE, NULL, NULL>)
OF
Contain(BAttr21, 'distributed database', <ANY_POSITION,
IS_PHRASE, NULL, NULL>)
```

In the SELECT clause, MarcToText() converts the MarcString for MAttr245 (title statement) to a normal string. Extract() extracts all the sub-elements with subtag '$a' from marc-attribute MAttr100.

Q2: Retrieve the title, subject and location information of the books authored by 'John Smith' from the NTU and the NUS libraries.

```
SELECT  MAttr245, MAttr650, Location
FROM    NTUandNUS_VBibTB
WHERE   Contain(BAttr1003, 'John Smith', <NULL, IS_NAME, NULL,
        NULL>)
```

Instead of performing similar searches against two BIB tables in the NTU and the NUS libraries, the above query is formulated against the virtual BIB table NTUand NUS_VBibTB defined upon these two imported BIB tables. In the WHERE clause, 'John Smith' is treated as a person name in the contain predicate.

Q3: Retrieve from the NTU library the titles, authors and subjects of book records found in RefTB.

```
SELECT  a.MAttr245, a.MAttr100, a.MAttr650
FROM    BibTB@NTU_DB: a, RefTB@RefDB: b
WHERE   Contain(a.BAttr4, b.Title, <FIRST_IN_SUBFIELD,
        IS_PHRASE, NULL NULL>)
        AND Contain(a.BAttr1003, b.Author, <NULL, IS_NAME, NULL,
        NULL>)
```

This query uses the information in RefTB to search the BIB table in the NTU library. HarpSQL allows two kinds of tables to be joined in one single query. In the search predicates, b.Title is treated as a phrase and the query requires it to be the first data in any of the sub-element of a a.BAttr4.

Q4: Retrieve those image processing related books found in BookTB of the publisher but not in the NTU library:

```
SELECT  Title, Author
FROM    BookTB@PubDB
```

```
WHERE   Subject = 'image processing'
MINUS
SELECT  MarcToText(a.MAttr245, 60), Extract(a.MAttr100, '$a', 0,
        40)
FROM    BibTB@NTU_LibDB
WHERE   Contain(BAttr21, 'image processing', <NULL, IS_PHRASE,
        NULL, NULL>)
```

Here, we illustrate that HarpSQL supports the set difference operation on two relations with one of them retrieved from a MARC database.

5. Conclusions

In this paper, we propose extensions to the relational model and SQL language to represent information found in remote bibliographic databases, specifically those accessible through the well-accepted Z39.50 information retrieval protocol. By modeling these bibliographic databases as relational tables, the application developers and some sophisticated end-users can now easily access these bibliographic databases similar to the standard SQL databases. The extended model also serves to integrate bibliographic databases and relational databases in a loosely coupled manner.

Our proposed SQL extensions support queries that involve one or more bibliographic databases, as well as queries that involve both bibliographic and relational databases. To achieve the integration of two querying paradigms, we have proposed new SQL predicates and functions. The concept of a virtual bibliographic table has also been introduced to allow broadcasting of queries to multiple bibliographic database servers, and merging their returned results. We also demonstrate the usefulness of the proposed extended relational model and its SQL language using a series of examples.

As part of an integrated digital library project called HARP [17,18], we have also prototyped a distributed query processor that supports HarpSQL queries over a collection of bibliographic and SQL databases. The prototype architecture is shown in Fig. 7. In our architecture, a user-friendly query frontend allows users to formulate their HarpSQL queries in a window-based environment. The distributed query processor consists of a query manager and a number of query agents one for each remote database. The query manager is responsible for processing HarpSQL queries submitted by the query frontend or any other digital library application. The query agents act as wrappers that support subqueries to different kinds of remote database servers which are members of the integrated digital library environment.

In the following, we list several topics that need to be addressed as future work:

- Modeling of other forms of data. So far, our proposed extended model and SQL focus on integrating bibliographic data and structured data. We will study how they can be further improved to model and query general semi-structured text and unstructured data. In the case of semi-structured data, we believe that our extended model and SQL can be combined with those proposed by Blake and others [10].
- Query optimization. With the new extensions to the query language, we need to examine various possibilities to optimize the queries on bibliographic and relational databases. Query optimization in our integrated environment will be difficult because the participating database servers are autonomous entities which may

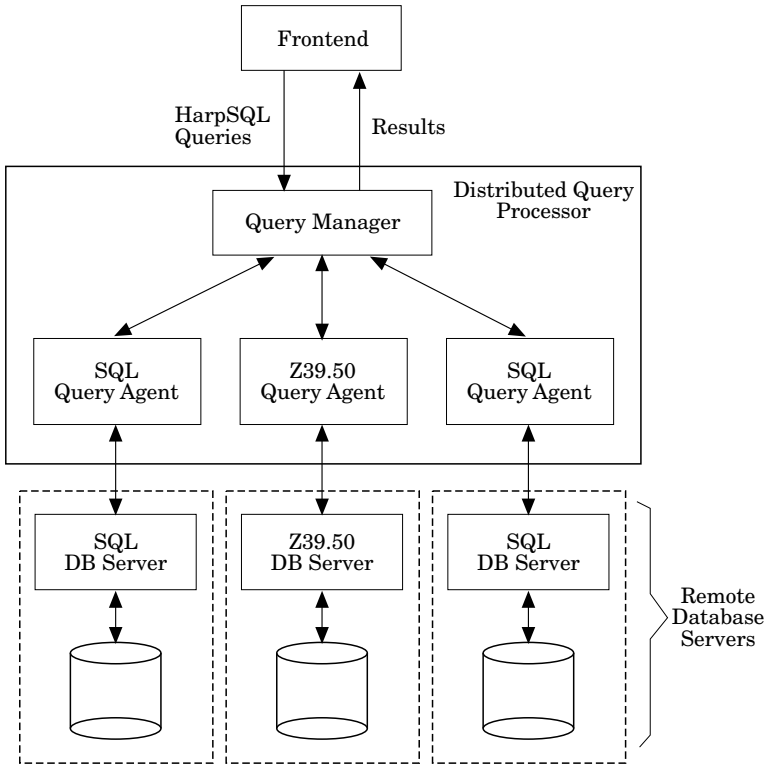

Figure 7.

adopt different local query processing strategies. Some optimization techniques for queries involving text and structured databases have been reported in [19].

- Duplication elimination. When queries are posed against a virtual bibliographic table, they are expected to return a large number of duplicate bibliographic records from different library systems. To prevent overloading the library applications or users with redundant data, we plan to develop algorithms to discard duplicated records in the results. Some related efforts in this area have been reported in [20, 21].

Acknowledgement

This work has been supported by the Nanyang Technological University-National Computer Board Memorandum of Understanding Research Grant.

References

1. B. Kahle and A. Medlar 1991. An information system for corporate users: Wide area information servers. *Connexions—The Interoperability Report*, **5**.
2. 1995. *Communications of the ACM*, **38(4)**.
3. S. J. Cannan and G. A. M. Otten 1993. *SQL—The Standard Handbook Based On the New SQL Standard (ISO 9075:1992(E))*. McGraw-Hill Book Company.

4. NISO 1992. *ANSI Z39.50: Information Retrieval Service and Protocol*. NISO Press.
5. W. Crawford 1984. *MARC for Library Use: Understanding the USMARC Formats*. Knowledge Industry Publications, Inc.
6. International Organization for Standardization 1986. *Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML), ISO/IEC 8879:1986*.
7. International Organization for Standardization 1989. *Information Processing—τext and Office Systems—Office Document Architecture (ODA) and Interchange Format (ODIF), ISO/IEC 8613:1989*.
8. S. C. Christophides, A. and M. Scholl 1994. From structured documents to novel query facilities. In *Proceedings of ACM SIGMOD Conference*, Minneapolis, MN. p 313–323.
9. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman and J. Wisdom 1994. The tsimmis project: integrating of heterogeneous information sources. In *IPSL Conference*, Tokyo.
10. G. E. Blake, M. P. Consens, I. J. Davis, P. Kilpelainen, E. Kuikka, P.-A. Larson, T. Snider and F. W. Tompa 1995. *Text/relational database management systems: overview and proposed sql extensions database prototype*. Technical Report 95–25, UW Centre for the New OED and Text Research, University of Waterloo.
11. D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman and J. Widom 1994. Querying semistructured heterogeneous information. Tecnical Report unpublished memorandum, Computer Science Dept., Stanford University.
12. W. Litwin and A. Abdellatif 1986. Multidatabase interoperability. *Computer*, 1986.
13. A. P. Sheth and J. A. Larson 1990. Federated database systems for managing distributed heterogeneous, and autonomous databases. *ACM Computing Surveys*, **22**.
14. ISO 1993. *Information Technology—Open Systems Interconnection—Remote Database Access (Part 1, Genetic Model) Service and Protocol*. ISO/IEC.
15. ISO 1993. *Information Technology—Open Systems Interconnection—Remote Database Access (Part 2, SQL Specialization)*. ISO/IEC.
16. D. Steedman 1990. *Abstract Syntax Notation One (ASN.1): the Tutorial and Reference*. Technology Appraisals.
17. E.-P. Lim, S-Y. Cheng and B-S. Lee 1995. Harp: an integrated digital library. In *Proceedings of International Symposium on Digital Libraries*, Ibaraki, Japan, August.
18. Ee-Peng Lim and S-Y. Cheng. Libsearch 1996. A window-based frontend to remote bibliographic databases on the internet. In *Third International Workshop on User-Interfaces to Database Systems*, Edinburgh, U.K.
19. S. Chaudhuri, U. Dayal and T. W. Yan 1995. Join queries with extended text sources: execution and optimization techniques. In *Proceedings of ACM SIGMOD Conference*, San Jose, CA, pp. 410–422.
20. M. J. Ridley 1992. An expert systen for quality control and duplicate detection in bibliographic databases. *Program*, **26**, 1–18.
21. T. W. Yan and H. Garcia-Molina 1995. *Duplicate removal in information dissemination*. Technical report, Computer Science Dept., Stanford University.

Appendix A: mapping between Bib-1 and MARC attributes

Bib-1 id	Name	MARC Tag(s)
62	Abstract	520
1003	Author	100, 110, 111, 400, 410, 411, 700, 710, 711, 800, 810, 800
1000	Author–title	100/2XX. 110/2XX, 111/2XX, 400, 410, 411, 700, 710, 711, 800, 810, 811
1005	Author–name corporate	110, 410, 710, 810
1006	Author–name conference	111, 411, 711, 811
1004	Author–name personal	100, 400, 700, 800
13	Dewey classification	082
16	LC call number	050
55	Code–geographic area	043
56	Code–institution	040
54	Code–language	088, 041
9	LC card number	010, 011
12	Local number	001, 035
30	Date	005, 008, 260, 033, etc.
31	Date of publication	008, 260, 046, 533
1011	Date/time added to database	008
1012	Date/time last modified	005
7	ISBN	020
8	ISSN	022, 4XX, 7XX
1007	Identifier–standard	010, 011, 015, 017, 018, 020, 022, 023, 024, 025, 027, 028, 030, 035, 037
1002	Name	100, 110, 111, 400, 410, 411, 600, 610, 611, 700, 710, 711, 800, 810, 811
57	Name and title	100/2XX, 110/2XX, 111/2XX, 400, 410, 411, 600, 610, 611, 700, 710, 711, 800, 810, 811
2	Corporate name	110, 410, 610, 710, 810
3	Conference name	111, 411, 611, 711, 811
58	Name geographic	651
1	Personal name	100, 400, 600, 700, 800
63	Note	5xx
21	Subject heading	600, 610, 611, 630, 650, 651, 653, 654, 655, 656, 657, 69X
24	INSPEC subject	600, 610, 611, 630, 650, 651
27	LC subject heading	600, 610, 611, 630, 650, 651
1008	Subject–LC children's	600, 610, 611, 630, 650, 651
1009	Subject name–personal	600
4	Title	130, 21X–24X, 400, 410, 440, 490, 600, 610, 611, 700, 710, 711, 730, 740, 800, 810, 811, 830, 840
43	Title abbreviated	210, 211, 246
37	Title added–title-page	246
38	Title caption	246
34	Title collective	243
36	Title cover	246
5	Title series	400, 410, 411, 440, 490, 800, 810, 811, 830, 840
6	Title uniform	130, 240, 700, 711, 730

Appendix B: search mode of contain( )†

(a) Position sub-mode

Option	Meaning
FIRST_IN_ELEMENT	Search term must be the first data in the element
FIRST_IN_SUBELEMENT	Search term must be the first data in the sub-element
ANY_POSITION*	Search term may appear at any place

(b) Structure sub-mode

Option	Meaning
IS_WORD	A word search term contains no blanks. It specifies the exact text of the value to be searched
IS_PHRASE*	A phrase search term consists of one or more words separated by blanks. It will be treated with respect to order and adjacency
IS_WORD_LIST	A word list search term consists of one or more words separated by blanks. No order of the words is implied
IS_NAME	The search term is treated as a person name
IS_STRING	The entire term is to be treated as a string, rather than a sequence or set of individual words

(c) Truncation sub-mode

Option	Structure option	Meaning
RIGHT_TRUNC*	Word/Phrase	Last word of term is right truncated
	String	Entire term is right truncated
	Word list	Each word is right truncated
LEFT_TRUNC	Word/Phrase	First word of term is left truncated
	String	Entire term is left truncated
	Word list	Each word is left truncated
LEFT_AND_RIGHT_TRUNC	Word/Phrase	First word of term is left truncated Last word of term is right truncated
	String	Entire term is left and right truncated
	Word list	Each word is left and right truncated
DO_NOT_TRUNC		No truncation is to be applied
PROCESS_#		The search term contains '#' to show where truncation will take place
REGULAR_EXP		The term is in the form of a regular expression

(d) Completeness sub-mode

Option	Meaning
INCOMPLETE_SUBELEMENT*	Words other than those in the search term may appear in the element/sub-element in which the term appears
COMPLETE_SUBELEMENT	No words other than those in the search term should appear in the sub-element in which the term appears
COMPLETE_ELEMENT	No words other than those in the search term should appear in the element in which the term appears

†For each sub-mode, the option marked with an asterisk is the default value.

Ee-Peng Lim received the B.S. (honours) degree in Computer Science from National University of Singapore in 1989, and Ph.D. degree in Computer Science from University of Minnesota, Minneapolis, MN in 1994. He is currently a lecturer in the School of Applied Science, Nanyang Technological University, Singapore, which he joined in 1994. He has authored over 20 refereed conference and journal papers in the areas of digital libraries, database integration, multidatabases, and query optimization. At present, he is leading an integrated digital library research project known as HARP. Dr. Lim is a member of the Association for Computing Machinery.

Ying Lu received a B.S. degree in Computer Science from the Tsinghua University, P.R.C. in 1994, and a master degree in Computer Engineering from Nanyang Technological University, Singapore in 1996. Presently, she is a software engineer in the Insititute of Systems Science (ISS), Singapore. Her research interests include digital libraries and content retrieval of multimedia objects.